cataloger Documentation

Release 0.0.1

Tony Flury

Mar 18, 2018

Contents

The cataloger is an easy to use tool to create a catalog of the contents of a directory tree, and then later use that same catalog to confirm that contents of the directory tree has been correctly copied/deployed. The catalog contains a hex-string signature created for each selected file within the directory tree, and the tools can identify files which been added, removed or are different (different content) within the directory tree.

CHAPTER 1

Selecting Files

The tool has a comprehensive set of tools for selecting which files are selected for inclusion in the catalog :

- selection by file extension
- de-selection of *top level directories*
- fine grained inclusion and exclusions of files based on glob pattern matching.

Information about the directory tree (directory and file names) are captured relative to a defined *root* when the catalog is created, which means that the absolute location of the directory tree on the platform is not relevant to the cataloger - what is relevant is the directory structure under the *root*.

For more details see *Command Line Options* and *Configuration file* for more information on how to select files for cataloging.

CHAPTER 2

Reportable Exceptions

The identification of additional files, missing files and files with different contents are termed to be *reportable anomalies*; and using command options exist to determine what type of reportable anomaly results are actually reported or are classed as failures; it is entirely possible therefore for the tool to be used in an environment where the additional files are acceptable for instance. See *Results and Reports* for more information.

CHAPTER 3

Defaults, Command lines and Configuration files

The catalog tool has a set of *default options* which are ideal for cataloging the deployment of a Django project, since Django projects have the same directory structure when deployed as they have on development (i.e. a hierarchical structure of a Project with multiple apps); but the catalog tool can be used in other environments as well by overriding the defaults file types, ignored directories and other options. See *Use Case* for more information.

If you use non default options to create your catalog, then you will need to ensure that you use the same options when you check your destination directory structure to ensure that the right files are cataloged; this is easily enabled by the use of a configuration file which holds all of the options used to both create and check the catalog; see *Configuration File* for more information on how to write these configuration files.

CHAPTER 4

APIs

Although primarily the catalog tool is provided as a command line tool, the software also includes APIs so that the cataloging functionality can be accessed from other python application. see *Catalog APIs* for more information on the two APIs provided.

Note: Every care is taken to try to ensure that this code comes to you bug free. If you do find an error - please report the problem on :

- GitHub Issues
- By email to : Tony Flury

4.1 Getting Started

4.1.1 Installation

Manifest checker can be installed using pip :

The cataloger tool and APIs are compatible with both Python 2 (2.7 and later) and Python 3 (3.5 and later)

4.1.2 Simple Usage

Using the cataloger command line tool with the default arguments is easy :

To create the manifest

```
$ cd <tree root>
$ catalog create
```

This will create a catalog.cat - detailing all the source code files in the current directory tree, with a sha224 hash created for each file. You can then deploy/copy your source code to the destination and execute

```
$ cd <tree root>
$ catalog check
```

By default, the check subcommand rescursively searches the directory tree under the current directory, and computes the hash for each file - and compares that against the computed has in the manifest file.

Note: Make sure you copy/deploy your catalog.cat file along with your source code - if you don't the `catalog check` command will fail immediately.

By default the `catalog check` will detect and report on three different types of issue :

- missing files : files listed in the manifest file, but which don't exist in the local directory tree.
- record_extra files : files which are not listed in the manifest file, but which exist in the local directory tree.
- incorrect hashes : files where there is a discrepancy between the hash created for the local file, and the hash listed in the manifest.

4.1.3 Further Reading

- Complete details of all the command line options : doc:*CommandLine Options*
- Details of how to write config files : doc:*config*
- Details of the default options that are applied : doc'Defaults'
- Detailed use cases of typical usage and suggested configurations : doc:*UseCases*

4.2 Command Line Options

4.2.1 Summary

The full command line options are:

```
manifest [-h] [-v, --verbose {0,1,2,3}]
                [-r,--root ROOT]
                [-o,--out REPORT-OUT]
                [-a,--hash {sha224,md5,sha1,sha256,sha384,sha512 - and possibly more}]
                [-m,--catalog MANIFEST]
                [-c, --config CONFIG]
                [-N, --no_config ]
                [-e, --rm_extension EXTENSION]
                [+e, --add_extension EXTENSION]
                [-d, --rm_directory DIRECTORY]
                [+d, --add_directory DIRECTORY]
                [-f, --exclude_filter FILTER]
                [+f, --include_filter FILTER]
                [-t/-T]
                [-k/-K]

        create

        check   [-m/-M ]
                [-i/-I ]
                [-e/-E ]
                [-g/-G ]
                [-s, --summary]
```

4.2.2 General options for all commands

-h, –help : Show the full help page and exit

-v, –verbose : The verbose reporting level from 0 or 1. The default is 1. Level 0 : No output, except execution error messages. Level 1 : Full output. Level 2 : Extended output for each directory.

General processing

-a, –hash HASH Choose a hash algorithm to be used to create the signature. Can at least be one of sha224, md5, sha1, sha256, sha384, sha512 the default is sha224. On a standard default installation at least the above algorithms listed will be available, and some packages - such as OpenSSL will make more algorithms available - Check the help page of the catalog suite on your installation to check what is available on your system. If you are using the catalog on two different systems ensure that you use a hash algorithm which is available on both.

This option is included for completeness - there is very little practical benefit to use anything other than sha224

-m, –catalog CATALOG The name of the catalog file to create or to check against. This option will create or use the catalog file with the specified name, rather than the default `catalog.cat`

-c, -config CONFIG Specify a config file to use, rather than the default `catalog.cfg` If a config file is specified which does not exist or cannot be opened a warning is generated to stderr, and execution continues using the *defaults values* only (modified by any command line options). If the -c option is not used and the default `catalog.cfg` does not exist or cannot be opened, then no warning is generated.

-N, –no_config A flag to suppress reading of the default config file `catalog.cfg`, and use just the *defaults values* only (modified by any command line options).

-r, –root ROOT The root directory to create the manifest from, or check the manifest against.

File Selection

-e, –rm_extension EXTENSION Remove one or more file extensions from the list of those to be cataloged. All the EXTENSION must start with a dot character otherwise an error will be raised.

+e, –add_extension EXTENSION Add one or more file extensions to the list of those to be cataloged. All the EXTENSION must start with a dot character otherwise an error will be raised.

-d, –rm_directory DIRECTORY Remove one or more directory from the list of those *top level directories* (relative to the root) which are ignored

+d, –add_directory DIRECTORY Add one or more directory from the list of those *top level directories* (relative to the root) which are ignored.

-f, –exclude_filter FILTER Add one or more glob filters to exclude files from being cataloged. If a file matches any of the exclude filters then it will not be cataloged or checked against the catalog.

+f, –include_filter FILTER Add one or more glob filters to include files into the catalog. If a file matches any of the include filters then it will be cataloged and checked

General Output Flags

-t/-T A flag to enable or disable counting and reporting of file extensions.

-l/-L A flag to enable or disable counting and reporting of excluded files. See *Counting excluded files* for more details

For these flags the lowercase option enables the report, and the uppercase option disables the report.

4.2.3 Check Command options

General options for the check command only - If specified these must occur after the *check* command :

Exception reporting flags

-m/-M Whether or not to report on mismatched files Those files where the file in the directory being checked has a different hashed signature compared to the signature recorded in catalog for that file

-i/-I

Whether or not to report on missing files Those files which exist in the catalog but do not exist within the directory structure being checked

-x/-X Whether or not to report on record_extra files Those files which exist within the directory structure being checked but do not exist in the catalog

For these flags the lowercase option enables the report, and an uppercase option disables the report.

If a particular exception type is disabled then no occurrence of this type of exception will not be reported and the command will not exit with a failure status. By default all of these reports are enabled.

4.2.4 Notes and Other Information

Relationship between System Defaults, Config file and command line options

With the exception of the –help option above all of the command line options have a system wide Default value and a counterpart option which can be used within a *config file*.

When identifying the values to be used for each option the following ordered process is executed :

1. The System wide defaults are applied in all cases

1. Assuming a config file exists (see -c option above) then the values and options from the config file are then applied either overridding or modifiying the values derived from the System wide defaults

1. Any options specified from the command line are then applied, further overiding or modifying the values generate previously.

Selection of files to be cataloged

Since there are multiple mechanisms for including and excluding files from the catalog it is worth exploring the rules in details to avoid confusion.

In strict order :

- All files within top level directories (as modified using the -d/+d options and the [directories] section of the config file) are not cataloged regardless of their file extensions
- Files whose files extension does not match one of the file extension list (as modified by -e/+e and the `[extensions]` section of the config file) are not cataloged.
- Files which match any specific exclusion filter (the -f option or the `exclude` option in the config `[filters]` section) are not cataloged.
- Files which do not match any specific included filter (the +f option or the `include` option in the config `[filters]` section) are not cataloged.

All other files are cataloged.

4.3 Results and Reports

4.3.1 Create command Report

By default a create command will generate a report to stdout similar to this :

```
12 files processed - 10 files excluded

Files processed by file types:
    .py : 6
    .txt : 3
    .rst : 3
```

Counting excluded files

The count of excluded files is :

- All files excluded due to a non cataloged file extension

- All files exluded due to matching a defined exclude filter
- All files exluded due to not matching a defined inculde filter

Files in *top level directories* (see -d/+d options) are not counted as being excluded.

Impact of verbosity settings

The above report is the default from the command line, when the verbosity of 1 is use. The is the default for the command line.

If verbosity = 0 (i.e. the command line option *-v 0* is used), then the above report is suppressed

if verbosity = 2 or 3 (command line option *-v 2*, or *-v 3*) then the report will also contain a table with a row for every directory; the table will count the number of files in that directory which are cataloged from that directory, and also a count of those excluded. The difference between the verbosity levels is that at verbosity level 2, directories are only included in the table if at least one file from that directory was included in the catalog. At verbosity level 3, all directories that were analyzed will be included in the table.

Note: The table for verbosity level 2 & 3

4.3.2 check command Report

By default a check command will generate a report to stdout similar to this :

```
12 files processed - 10 files excluded

Files processed by file types:
    .py : 6
    .txt : 3
    .rst : 3
0 files with mismatched signatures
0 missing files
0 extra files
```

Impact of verbosity settings

The above report is the default from the command line, when the verbosity of 1 is use. The is the default for the command line.

If verbosity = 0 (i.e. the command line option *-v 0* is used), then the above report is suppressed

if verbosity = 2 or 3 (command line option *-v 2*, or *-v 3*) then the report will also contain a table with a row for every directory; the table will count the number of files in that directory which are cataloged from that directory, and also counts of mimatched, missing, extra, and excluded files. The difference between the verbosity levels is that at verbosity level 2, directories are only included in the table if at least one file from that directory was included in the catalog. At verbosity level 3, all directories that were analyzed will be included in the table.

4.3.3 Command line exit status

When the check command is executed then command will exit with a status of 0 when there are no *reportable anomalies*, and a status of 1 when at least one *reportable anomalies* exists.

if this means it is entirely possible to use the check command in a bash script or similar :

```
if ! catalog check >check_output ; then
    echo 'Integrity check failed :'
    more check_output
    exit 1
```

4.4 Defaults

By default the manifest suite with the default options as above will create a manifest for a Django projects (typically these projects exist as a nested set of directories which containt various types of source code, including python css, javascript and html files (and templates), as well as image files (gif, jpeg, png etc). A Django project will also contain directories and files which aren't part of the deployment, which don't need to be included in the integrity check by default.

If you don't use Django, you can still use the manifest suite to check your copies/deployments. Various options command line and Configuration File options exist to make the manifest suite usable including :

- check for different files types
- ignoring different directories
- filtering to include and exclude file based on wildcard matches

File Extension .py, .html, .txt, .css, .js, .gif, .png, .jpg, .jpeg

Top Level directories which are ignored : static, env, htmlcov, media,build, dist, docs

Default manifest file 'catalog.cat'

Default hash/checksum algorithm. 'sha224' if 'sha224' in hashlib.algorithms else hashlib.algorithms[0]

Reporting options skipped files and file extension totals are not reported on, but they can be included by using -k and -t options on the manifest command.

Checking options mismatched, missing and record_extra files are included in the reports. These checks can be disabled by using the -M, -I and -X options respectively.

4.5 Configuration file

By default the config file is named 'catalog.cfg' and if that file exists the configuration options there are used before those specified on the command line. The values specified in the config file are used for both create and check commands; and thus using a config file can be very useful in ensuring consistency without typing complex command lines.

4.5.1 Sections

The config file has a number of sections - none are mandatory - although an empty config file is pointless.

Each section has the following format :

```
[title]

<optionline>
```

Sections can contain one or more <*optionline*>, and the exact form of these option lines depends on the section.

Leading and trailing whitespace on all lines within the section is ignored.

Lines within the config file that begin with a # are ignored - and can be used as comments

catalog Section

Note: This section is equivalent to the *-m/–catalog*, *-h/–hash* and *-r/–root* command line options

This section configures the general options related to creation of the catalog

The format of this section is :

```
[catalog]

catalog = <file name>
hash = <hash name>
root = <directory path>
```

where the `<option>=<value>` line can be provided for each option - if an option is repeated then the last value given is used. Options can be omitted in which case the system default for that option is used.

Available options are :

catalog The name of the catalog file. Equivalent to the `-m/--manifest` command line option. If not provided defaults to `catalog.cat`

hash The name of the hash algorithm to use. The name does not need quotes. Equivalent to the `-h/--hash` command line option. Defaults to using sha224 hash.

root The root directory to use - so that all files under the root will be analysed and catalogued. Equivalent to the `-r/--root` command line option. This can be either a relative or absolute path (although it makes more sense to be relative) Defaults to '.'

Spaces and tabs around the = are optional.

Extensions Section

Note: This section is equivalent to the +e/–add_extension and -e/–rm_extension command line options

This section configures the files extensions which are used when selecting files to be cataloged.

The format of this section is :

```
[extensions]

= <new extension list>
+ <additional extension list>
- <remove extension list>
```

The section can have one or more of each possible type of option line - and they can appear in any order. These lines are processed in order to build an list of file extensions which are to be cataloged or checked.

= <new extension list> Reset the extension list from it's current value to just those items in the `<new extensions list>`; There is no command line equivalent for this option. The extension list can be omitted from this option, in which case the extension list is emptied.

+ <additional extension list> Add those extensions listed in `<additional extension list>` to the extension list to be used; Equivalent to a one or more `+e` command line options

- <remove extension list> Remove the extensions list in the `<remove extension listt>` from the extension list to be used; Equivalent to a one or more `-e` command line options.

Each of these extension lists are comma separated, with each file extension in the list starting with a `.` (a dot). Spaces and tabs around the individual file extensions are ignored.

Spaces after =, +, and − are optional

Lines in this section which start with any character other than =, +, − and # will be reported as an error.

Warning: Within this section ordering matters. The lines are executed strictly in the order they appear; so for instance :

- Using the + and − operators followed by an = operator will result in the changes implemented by the + & − operators being pointless. An error will not be raised.
- Having multiple =' operations within the section will result in only the latest one having an effect. An error will not be raised.

Examples

Using the + operator

Assuming that the default extension list is `.py, .html, .txt, .css, .js, .gif, .png, .jpg, .jpeg` then :

```
[extensions]

+ .htm, .cfg, jsx
```

Would result in the extension list of `.py, .html, .htx, .txt, .css, .cfg, .js, .jsx, .gif, .png, .jpg, .jpeg` being used for both create and check operations.

Note: It is not an error to use the + to add an extension that already exists in the current list.

Using the - operator

Assuming that the default extension list is `.py, .txt, .html, .css, .js, .gif, .png, .jpg, .jpeg` then :

```
[extensions]

- .html, .js, .css
```

Would result in the extension list of `.py, .txt, .gif, .png, .jpg, .jpeg` being used for both create and check operations.

Note: It is not an error to use the − to attempt to remove an extension that doesn't exist in the current list.

Using the = operator

Assuming that the default extension list is `.py, .txt, .html, .css, .js, .gif, .png, .jpg, .jpeg` then :

```
[extensions]

= .html, .js, .css
```

Would result in only files with extensions of `.html, .js` and `.css` only being used for both create and check operations.

Directories Section

Note: This section is equivalent to the -d/–rm_directory and +d/–add_directory command line options

This section configures which *top level directories* are to be excluded from the catalog.

The format of this section is :

```
[directories]

= <new directory list>
+ <additional directory list>
- <remove directory list>
```

The section can have one or more of each possible type of option line - and they can appear in any order. These lines are processed in order to build an list of file extensions which are to be cataloged or checked.

= <new directory list> Reset the directory list from it's current value to just those items in the `<new directory list>`; There is no command line equivalent for this option. The directory list can be omitted from this option, in which case the directory list is emptied.

+ <additional directory list> Add those directory listed in `<additional directory list>` to the directory list to be used; Equivalent to a one or more `+d` command line options

- <remove extension list> Remove the directory list in the `<remove directory listt>` from the directory list to be used; Equivalent to a one or more `-d` command line options.

Each of these directory lists are comma separated, Spaces and tabs around the individual file extensions are ignored.

Spaces after =, +, and − are optional

Lines in this section which start with any character other than =, +, − and # will be reported as an error.

Warning: Within this section ordering matters. The lines are executed strictly in the order they appear; so for instance :

- Using the + and − operators followed by an = operator will result in the changes implemented by the + & − operators being pointless. An error will not be raised.

- Having multiple =' operations within the section will result in only the latest one having an effect. An error will not be raised.

Examples

Using the + operator

Assuming that the default directory list is **static**, **env**, **htmlcov**, **media**, **build**, **dist**, **docs** then :

```
[extensions]

+ .tox
```

Would result in the top level directories **static**, **env**, **htmlcov**, **media**, **build**, **dist**, **docs** and **.tox** being excluded from the catalog.

Note: It is not an error to use the + to add an directory that already exists in the current list.

Using the - operator

Assuming that the default extension list is **static**, **env**, **htmlcov**, **media**, **build**, **dist**, **docs** then :

```
[extensions]

- media, docs
```

Would result in the in the *top level directories* **static**, **env**, **htmlcov**, **build**, **dist**, and **.tox** only being entirely excluded from the catalog, with the contents of the **media** and **docs** directories being cataloged, subject to file extensions and filters

Note: It is not an error to use the − to attempt to remove an director that doesn't exist in the current list.

Using the = operator

Assuming that the default extension list is **static**, **env**, **htmlcov**, **media**, **build**, **dist**, **docs** then :

```
[extensions]

= static, htmlcov
```

Would result in only files with extensions of **static**, **htmlcov** only being entirely excluded from the catalog, with the content of all other top level directions being cataloged, subject to file extensions and filters.

Filter Section

Note: This section is equivalent to the -f/–exclude_filter and +f/–include_filter command line options

The format of this section is :

```
[filters]

include = <glob pattern match list>
exclude = <glob pattern match list>
```

The glob pattern match list is used as the full set of glob pattern matches to be used when identifying which specific files are to be cataloged. The normal rules of glob matching applies :

> ? matches any single character - for instance *?ython* will match *Python* as well as *ython* * will match zero or more of any character - for instance **ython* will match *ython*, *Python*, and *ython*

The include and exclude patterns are applied to the individul files names (relative to the root directory) and not the directory names - so to specifically match against a directory and all of its contents (say the directory *templates*), it is recommended to use a form : **templates**

Note: Files within the top level directory are cataloged with file names starting with *./*, and all other files are cataloged as file paths relative to the root, but without the './'

Note: When selecting which files to catalog, any exclusion filters are applied first, and if the full file path matches any single exclusion filter it is not cataloged (regardless of it's file extension or whether it matches any inclusion filter): See *Selection of files to be cataloged* for a full description of how files are chosen for cataloging.

Reports Section

Note: This section is equivalent to the -v/–verbose, -k/-K, -t/-T, -m/-M, -i/-I, -x/-X command line options

The report section format is :

```
[reports]

verbose = <level>
mismatch = <flag>
missing = <flag>
extra = <flag>
excluded = <flag>
extension = <flag>
```

The options are :

verbose Equivalent to the *-v/–verbose* option; defines the verbosity level of the output. <level> is a numeric value one of 0, 1 or 2. Level 0 supresses all text output. Level 1 (the default) produces the final report only. Level 2 produces a summary for each directory where there are *reportable anomalies*, as well as the final report; applies on both *check* and *create* commands

mismatch Whether or not to report on mismatched files (i.e. those files which have a diferent hash signature in the directory structure than is recorded in the catalog); equivalent to the *-m/-M* command line option; only applies on the `check` command

missing Whether or not to report on missing files (i.e. those files which are recorded in the catalog but which don't exist within the directory structure); equivalent to the *-i/-I* command line option; only applies on the `check` command

extra Whether or not to report on extra files (i.e. those files which exist within the directory structure, but which are not recorded in the catalog); equivalent to the *-x/-X* command line option; only applies on the `check` command

excluded Whether or not to report on the number of files which are excluded (i.e. all files excluded due to a non cataloged file extension [see *Extensions Section*] and all files exluded due to matching a defined exclude filter [see *Filter Section*] and all files exluded due to not matching a defined inculde filter [see *Filter Section*]; equivalent to the *-l/-L* command line option, applies on both *check* and *create* commands

extension Whether or not to report on counts per file extensions which are cataloged (i.e. those files which either cataloged on a *create* command or checked on the *check* command); equivalent to the *-t/-T* command line option, applies on both *check* and *create* commands

For the mismatched, missing, extra, excluded, extension options :

> <flag> is a representation of a boolean value - a value of *True* or *Yes* (or any upper or lower case version of those - so for example *True*, *true*, *tRuE*, *yes*, *Yes* and *yeS* all qualify as a representation of *True*. Any value of False or No (or any upper of lower case version of those - for example *False*, *false*, *fAlSe* or *No*, *no*, *NO* will qualify as a representation of 'False').

4.6 Typical Use Case

The cataloger was designed initially to verfiy the deployment of of a Django based web site. When Using the Django framework the directory structure in the development environment (that of a project directory containing one or more app directories) is the same directory structure replicated in when the project is deployed; the same files in the same relative places - with a few exceptions - for instance in Django you may well not deploy the contents of your Static content directory from your development environment).

There are a number of production/live environments which use git/git hub to transfer between development and deployment - the normal process would be :

On the development machine :

1. Develop
2. Test
3. git commit
4. git push

On the deployment platform :

1. Clone from github
2. Setup
3. Deployment tests
4. enable

The challenge with this process can be that there is no confirmation that all the neccessary files have been added to your git repo, so the content on the deployment platform maybe missing key files, or a file gets inadvertantly omitted from the latest committ, and therefore the clone on the deployment server fails to replace some files which were actually updated during development - no amount of testing on the development server will identify files missing from the github repo.

With cataloger, the development process becomes :

On the development machine :

1. Develop
2. Test
3. catalog create
4. add catalog file and catalog.cfg to git repo
5. git commit
6. git push

On the deployment platform :

1. Clone from github
2. catalog check - using the cloned catalog.cfg
3. Setup
4. Deployment tests
5. enable

With this revised process, you know by step 2 on the deployment platform if there are missing, extra or non-updated files within the deployment platform; it is very literally a confirmation that what you are deploying is what you have tested - without even have to run any potentially expensive tests.

4.7 Cataloger API

The cataloger package supports a fully featured programmatic API which allows Python scripts to build catalogs and check the directory structure against a previously built catalog.

4.7.1 Exceptions

exception `cataloger.processor.`**`CatalogError`**

Raised when there is an error within the catalog file itself (i.e. a formatting issue).

exception `cataloger.processor.`**`ConfigError`**

Raised when there is an error within the *config file*.

4.7.2 Command Methods

`cataloger.commands.`**`create_catalog`**`(`*verbose=0*, ***kwargs*`)`

`cataloger.commands.`**`check_catalog`**`(`*verbose=0*, ***kwargs*`)`

Using the directory structure, either create a catalog, or compare the directory structure against a cataloge. By default using settings defined in catalog.cfg if it exists. If a setting is defined in the *config file* it can be overidden by argumensts passed to the function.

Returns a `Cataloger` instance.

Parameters

- **`no_config`** (*`Boolean`*) – True if the config file should be ignored. Defaults to False
- **`config`** (*`str`*) – The name of the config file to use Defaults to ‘’. Unless a specific config file is provided the processor will attempt to read defaults.DEFAULT_CONFIG_FILE as the config file.
- **`catalog`** (*`str`*) – The name of the catalog file to use Defaults to catalog.cat
- **`hash`** (*`str`*) – The name of the hash algorithm to use Defaults to sha224
- **`root`** (*`str`*) – The root directory to start the creation or check process. Can be either a absolute or a path relative to the current working directory Defaults to ‘.’
- **`extensions`** (*`set`*) – The file extensions to catalog. Defaults to .py, .html, .txt, .css, .js, .gif, .png, .jpg, .jpeg
- **`rm_extension`** (*`set`*) – A set of extensiions to remove from catalogue No Default
- **`add_extension`** (*`set`*) – A set of extensiions to add to catalogue No Default
- **`ignore_directory`** (*`set`*) – A set of directories under root which are to be ignored and not catalogued Defaults to static, htmlcov, media, build, dist, docs
- **`rm_directory`** (*`set`*) – A set of directories to removed from the ignore_directory set. No Default
- **`add_directory`** (*`set`*) – A list/set of directories to be added to the ignore_directory set. No Default
- **`include_filter`** (*`list`*) – A glob file matching filter of files to catalogue. Default behaviour is that all files which have a file extension in the `extensions` set are catalogued
- **`exclude_filter`** (*`list`*) – A glob file matching filter of files to exclude from catalogue. Default behaviour is that no files which have a file extension in the `extensions` set is excluded from the catalogue.

Raises

- ***`processor.CatalogError`*** – If an error exists within the catalog file itself (or it cannot be read).
- ***`processor.ConfigError`*** – If an error exists within the config file itself.

4.7.3 Config file processing :

All of the arguments (and by extension the command line arguments) are processed after the config file if any.

If the *no_config* flag is True, then all *config* files are ignored and only the parameters passed to the functions are used.

If *no_config* is False (the default), and *config* is ‘’ or None, then the default config file is used only if it exists, and no error is created if the file doesn’t exist.

If *no_config* is False, and *config* is provided (even if it is the default name) then the config file is used if it exists, but a warning is generated if the file doesn't exist - execution continues as if the config file is empty.

4.7.4 Cataloger class

class `cataloger.commands.`**`Cataloger`**

An instance of the `Cataloger` class is returned by both `create_catalog()` and `check_catalog()`. The `Cataloger` class is not intended to be instantiated on it's own.

The `Cataloger` class contains a number of attributes and methods to enable programatic access to the results of the catalog creation or catalog checking tasks.

`catalog_file_name`
The read only name of the catalog file created or being checked.

`processed_count`
A read only count of the number files in the catalog.

`extension_counts`
A read only dictionary of file extensions and the count for each extension.

- key : file extension (with leading dot)
- value : A count of the files within the catalog with this extension

`excluded_files`
A read only list of the paths of all *excluded files*. All file paths are relative to the *root* path parameter.

`mismatched_files`
A read only list of the paths of all *mismatched files*. All file paths are relative to the *root* path parameter.

`extra_files`
A read only list of the paths of all *extra files*.All file paths are relative to the *root* path parameter.

`missing_files`
A read only list of the paths of all *missing files*. All file paths are relative to the *root* path parameter.

`catalog_summary_by_directory`

A generator method which yields a dictionary for each directory within the catalog - the dictionary has the following keys :

- path: The relative path of the directory being reported on
- processed: The count of all files in the catalog in this directory
- excluded: The count of all *excluded files* from this directory
- mismatched: The count of the *mismatched files* from this directory. Will always be zero after a `create_catalog()` call.
- missing: The count of the *missing files* from this directory. Will always be zero after a `create_catalog()` call.
- extra: The count of the *extra files* from this directory. Will always be zero after a `create_catalog()` call.

`is_file_in_catalog`(*file_path*)
True if this file exists in the catalog

`is_directory_in_catalog`(*directory*)
True if this directory exists in the catalog

4.8 Glossary

catalog A simple flat data store constructed by the *create sub-command*. The catalog provides a file path (stored as a path relative to the *root* directory) for each *selected file* within the directory structure, and a hash signature using the specified hash algorithm (sha224 is the default). The catalog is constructed recursively from the *root* downwards.

check The sub command to scan the local directory structure are create a report comparing the local directory structure with the previously *created catalog*. The check command uses the data in the catalog to identfy *mismatched* files, *extra* files and *missing* files.

create The sub-command to scan a directory structure and create a catalog of relevant files. The catalog contains the path to the file, and a signature hash value so that any subsequent changes can be reported.

extra A file which exists within the directory structure but does not exist within the catalog. The command line options -x/-X can be used to control if extra file anomalies are reported on.

mismatch Any file where the file within the directory structure has a different hash signature than the signature that exists in the catalog for the same file. The command line options -m/-M can be used to control if mismatch anomalies are reported on.

missing A file which exists in the catalog but does not exist within the directory structure. The command line options -i/-I can be used to control if missing file anomalies are reported on.

reportable anomaly Anomalies are the potential results of the *check* command. There are three types of anomaly : *mismatch*, *missing* and *extra* files.

root An optional argument provided to the command line or the API to identify where the analysis of the directory structure should start. It defaults to '.' (i.e. the current working directory when the *create* or *check* sub-commands or APIs are executed.

selected file A file which is identified as needing to be cataloged, or needing to be checked against the catalog.

There are multiple mechanisms for including and excluding files from the catalog and they are applied In strict order :

- All files within *top level directories* (as modified using the -d/+d options and the [directories] section of the config file) are not cataloged regardless of their file extensions
- Files whose files extension does not match one of the file extension list (as modified by -e/+e and the `[extensions]` section of the config file) are not cataloged.
- Files which match any specific exclusion filter (the -f option or the `exclude` option in the config `[filters]` section) are not cataloged.
- Files which do not match any specific included filter (the +f option or the `include` option in the config `[filters]` section) are not cataloged.

top level directory A directory contained within the *root* directory. The catalog tool set and APIs has the feature of being able to entirely disregard named top-level directories without analyzing the contents.

Python Module Index

Index